Lab 4

1. Determine whether InsertionSort, BubbleSort, SelectionSort from lesson 3 are *stable* sorting algorithms, and in each case, explain your answer.

2. Perform the MergeSort algorithm by hand on the array [7, 6, 5, 4, 3, 2, 1]. Show all steps.

3. Sometimes MergeSort is supplemented with a secondary sorting routine (typically, InsertionSort is used) in the following way: During the recursion in MergeSort, the size of the array being sorted becomes smaller and smaller. To create a hybrid sorting routine, when a recursive call requires the algorithm to process an array with 20 or fewer elements, give this array to InsertionSort and patch in the result after it has finished. Call this hybrid algorithm MergeSortPlus.

    A. Express the steps of MergeSortPlus in the pseudo-code language we are using in class.
    B. Write the Java code for MergeSortPlus (use the implementation of MergeSort provided in the lab folder)
    C. Run tests to compare running times of MergeSort and MergeSortPlus. Which one runs faster? Explain how you tested and whether you feel your results are conclusive.

4. *Binary Trees.* A *binary tree* is a tree in which every node has at most two children.

    a. Write out 4 different binary trees, each having height = 3 – make sure that no two of your trees have the same number of nodes. (There is no need to give labels to the nodes.)

    b. Examine the trees you have drawn and decide whether the following statement is true or false:

        Every binary tree of height 3 has at most $2^3=8$ leaves.

    c. Based on your answer to b, what do you think is true in general about the number of leaves of a binary tree of height *n*?

5. Solve the following problem with a recursive algorithm: Given a list with n elements, put the elements of the list in reverse order. Compute the running time of your algorithm (hint: count self-calls).